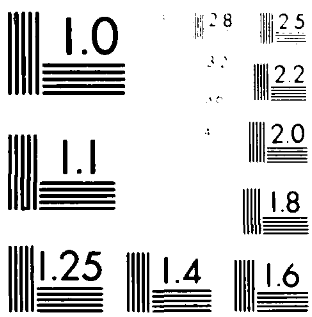
AD-A109 833 GENERAL ELECTRIC CO ARLINGTON VA INFORMATION SYSTEMS--ETC F/G 9/2
THE EFFECTS OF THE SYMBOLOGY AND SPATIAL ARRANGEMENT OF SOFTWAR--ETC(U)
DEC 81 S B SHEPPARD, J W BAILEY, E KRUESI N00014-79-C-0595
UNCLASSIFIED TR-81-388200-5 NL

1 OF 1
AD A
109833

END
DATE
FILMED
02-82
DTIC



MICROCOPY RESOLUTION TEST CHART

LEVEL II

12

THE EFFECTS OF THE SYMBOLOGY AND SPATIAL ARRANGEMENT OF SOFTWARE DOCUMENTATION IN A MODIFICATION TASK

AD A109833

SYLVIA B. SHEPPARD
JOHN W. BAILEY
ELIZABETH KRUESI

Software Management Research
Information Systems Programs
General Electric Company
1755 Jefferson Davis Highway
Arlington, Virginia 22202

TR-81-388200-5
DECEMBER 1981

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A109 833

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
The Effects of the Symbology and Spatial Arrangement of Software Documentation in a Modification Task

5. TYPE OF REPORT & PERIOD COVERED
Technical Report

6. PERFORMING ORG. REPORT NUMBER
TR-81-388200-5

7. AUTHOR(s)
Sylvia B. Sheppard, John W. Bailey, Elizabeth Kruesi

8. CONTRACT OR GRANT NUMBER(s)
N00014-79-C-0595

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Information Systems Programs
General Electric Company
1755 Jefferson Davis Hwy., Arlington, VA 22202

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
NR 196-160

11. CONTROLLING OFFICE NAME AND ADDRESS
Engineering Psychology Programs, Code 442
Office of Naval Research
Arlington, Virginia 22217

12. REPORT DATE
December 1981

13. NUMBER OF PAGES

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
same

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

same

18. SUPPLEMENTARY NOTES

Technical Monitor: Dr. John J. O'Hare

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software engineering, Software experiments, Structured programming, Modern programming practices, Software documentation, Flowcharts, Program design language, Software human factors, Software specifications

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the fourth in a series of experiments to evaluate the effects of the format of software documentation on programmer performance. The current experiment examined performance on a modification task. Thirty-six professional programmers were presented with documentation for each of three modular-sized programs. Nine different documentation formats were prepared for each program. These formats varied along two dimensions: type of symbology and spatial arrangement. The type of symbology included

normal English, abbreviated English and program design language (PDL). The spatial arrangement included sequential (vertical flow), branching (flowchart), and hierarchical (tree-like).

The participants were required to implement a modification to each program. These modifications required a minimum of three to five lines of additional code which the participants inserted using a text editor. The program output was checked automatically and a message informed the participants whether the output was correct or incorrect. The participants were asked to continue working until the modification was completed successfully. The difficulty of the task was measured by the time required to successfully complete the modification and by the number of errors which appeared in the first submission of the modified program.

Unlike the previous experiments in this series, the type of symbology did not have a strong effect on performance time. However, the results reflected the trend that appeared in the previous experiments: the more succinct the symbology, the better the performance.

Spatial arrangement did produce a strong effect in this experiment. The branching arrangement was associated with shorter performance times than the other arrangements. As in previous experiments, the participants preferred the most succinct symbology, the PDL, and the branching spatial arrangement.

TR-81-388200-5

THE EFFECTS OF THE SYMBOLOGY AND SPATIAL ARRANGEMENT OF SOFTWARE DOCUMENTATION IN A MODIFICATION TASK

SYLVIA B. SHEPPARD
JOHN W. BAILEY
ELIZABETH KRUESI

Software Management Research
Information Systems Programs
General Electric Company
1755 Jefferson Davis Highway
Arlington, Virginia 22202

Submitted to:

Office of Naval Research
Engineering Psychology Programs
Arlington, Virginia

A

Contract: N00014-79-C-0595
Work Unit: NR 196-160

DECEMBER 1981

TABLE OF CONTENTS

INTRODUCTION

The success of any software development project depends in part on the quality of the communication among the individuals involved: users, designers, coders and managers. This is a particularly critical factor in the development of a large system since a variety of individuals perform various tasks at different points in time. The efficiency with which later tasks are performed depends critically on the documentation supplied during previous phases of the development cycle. Thus, both managers and programmers alike are interested in the relative merits of the many types of documentation currently in use. Included among these are English descriptions, flowcharts and program design languages (PDLs).

There have been several empirical investigations of the relative value of these different types of documentation. For example, in a study comparing flowcharts and PDL, Ramsey, Atwood, and Van Doren (1978) found no difference in the ease with which these two types of documentation could be comprehended; they did, however, find an advantage for PDL as a design tool. (For a summary of relevant studies, see Sheppard, Bailey, and Kruesi, 1981.)

In general, there are two primary dimensions for categorizing how available documentation aids configure the information they present to programmers (Jones, 1979). The first dimension is the type of symbology in which information is presented. The second dimension is the spatial arrangement of this information. PDL, for example, uses constrained language or pseudo-code as the symbology presented in a sequential spatial arrangement. Flowcharts use ideogram symbols presented in a branching spatial arrangement. As a consequence of the fact that documentation formats vary along more than one dimension, there is a limit to the conclusions that can be drawn from a comparison between two formats since such a comparison may not allow us to determine the source of any observed difference. For example, in the Ramsey et al. study cited above, the difference between PDL and flowcharts may be due to the differences in the symbols, in the spatial arrangement or to an interaction of these two dimensions.

Our approach to evaluating various forms of documentation is to investigate the separate and combined effects of the type of symbology and the spatial arrangement. By expanding our realm of study beyond a comparison of only two formats, we hope to discover more general principles which will aid software developers in selecting from the many available documentation aids as well as guide in the development of new aids.

The current experiment is the fourth in a series. In each experiment, three types of symbols were factorially combined with three spatial arrangements to produce nine different formats.

Type of Symbology

In Experiments 1, 2 and 3, the three types of symbology consisted of normal English, PDL and ideograms. Normal English is frequently used as a documentation tool. PDL, which is less verbose than normal English, uses strictly defined keywords to describe arguments or predicates. Ideograms are frequently found in flowcharts and HIPO charts; a standard set of ideograms has come to represent processes or entities within a program. In Experiment 4, the ideogram symbols were replaced by an abbreviated natural language. The reason for this substitution is explained below.

Spatial Arrangement

In all four experiments, the spatial arrangements were sequential, branching, and hierarchical. A sequential arrangement is typical of narrative descriptions, program listings and PDL, while a branching arrangement is typical of flowcharts. A hierarchical arrangement is not generally used for individual module specifications but is used at the system level to present a visual display of the relationship among modules.

The results of the first three experiments are described briefly in the following sections. The first experiment, which is described in Sheppard, Kruesi, and Curtis (1981), investigated the influence of these dimensions on comprehension performance. The second experiment examined the performance of programmers as they translated the various documentation formats into code (Sheppard and Kruesi, 1981) while the third experiment examined performance on a debugging task.

Effects of Symbology and Spatial Arrangement on Comprehension

In the first experiment (Sheppard, Kruesi & Curtis, 1981), seventy-two professional programmers were presented with documentation for each of three modular-sized computer programs. The participants answered a series of comprehension questions for each program using only the documentation (i.e., they were not given the actual program listing). The questions were presented interactively on a CRT and consisted of three different types. For forward-tracing questions, the participants were given the values for a set of conditions in the program. Their task was to trace through the documentation and find the first statement executed under those conditions. For backward-tracing questions, they were reqired to locate a given statement within the documentation and then determine the set of conditions which led to that point. For the input-output questions, they were given input data and were asked to determine the value of particular variables at a later point in the program.

Both forward and backward-tracing questions were answered more quickly from documentation presented in PDL or ideograms than in normal English. On the average, forward-tracing questions were answered most quickly from a branching arrangement and backward-tracing questions were answered more quickly from the branching and hierarchical arrangements. An examination of the individual formats revealed that the sequential PDL, the branching PDL and the branching ideogram versions were associated with very quick responses for both types of questions. For the input-output questions, no significant differences were found as a function of the type of symbology or the spatial arrangement. At the conclusion of the experimental session, participants were asked to list the type of symbology and the spatial arrangement they most preferred. PDL was the most preferred symbology and the branching spatial arrangement was the most preferred arrangement.

Effects of Symbology and Spatial Arrangement in a Coding Task

In the second experiment (Sheppard & Kruesi, 1981), thirty-six professional programmers were presented with documentation and partially completed code for the same three programs. The participants constructed a major section of code at the middle of each program. About fifteen lines were missing from the code. This section included the most complex decision structures present in the program.

Substantial differences in performance were associated with the type of symbology. Coding from the normal English formats took considerably longer (29.7 minutes) than coding from the PDL (20.5 minutes) or ideogram (23.9 minutes) versions. An examination of the error data showed a similar pattern: the normal English formats resulted in a mean of 2.4 errors, the PDL resulted in 0.8 error and the ideograms resulted in 1.4 errors.

The effect of spatial arrangement was not as great as the effect of symbology. Although not statistically significant, the branching arrangement appeared to be superior to the sequential and hierarchical arrangements, particularly in minimizing errors related to the control flow. A comparison of the individual formats revealed that the sequential PDL and the branching PDL resulted in the highest level of performance. The branching ideograms and the hierarchical ideograms were also associated with good performance. Of the nine formats, the sequential normal English version resulted in the lowest level of performance.

The participants' preferences for symbology and spatial arrangement were consistent with the time and error data. PDL was the symbology preferred most often and branching was the most preferred spatial arrangement.

Effects of Symbology and Spatial Arrangement in a Debugging Task

In Experiment 3 (Sheppard, Bailey & Kruesi, 1981), 36 professional programmers were asked to compare error-seeded program code to the same documentation formats in order to detect and correct the errors. There were three errors per program. These errors were selected from among those made during the coding task in Experiment 2. The participants were told that the errors were located in the center section of the programs but they were not told how many errors occurred in each program. The dependent variable was time to debug.

Again, substantial differences in performance were associated with the type of symbology. Debugging from normal English took longer (18.7 minutes) than debugging from either PDL (14.5 minutes) or ideograms (14.2 minutes). The overall effect of spatial arrangement was not pronounced. A comparison of the individual formats revealed that the sequential and branching PDL again led to a high level of performance as did the branching and hierarchical ideograms. The sequential normal English again resulted in very poor performance.

The participants had no preferences for the type of symbology but did prefer the branching spatial arrangement to the sequential and hierarchical arrangements.

Experiment 4 - Modification

In the first three experiments, normal English resulted in substantially longer response times than the other two symbologies. It appeared likely that at least part of this difference was due to the manner in which variable names were expressed. The normal English contained an English description of each variable while the PDL and ideograms contained the variables as they were used in the FORTRAN code. Thus, the normal English required more translation from the documentation to the code.

In Experiment 4, an abbreviated English was substituted for the ideograms in order to assess the extent to which the variable names account for the symbology effect. The abbreviated English was identical to the normal English with the exception that the variable names were used rather than normal English descriptions. Thus, the abbreviated English was more succinct than the natural language but less succinct than the PDL.

The task in Experiment 4 was to modify the three programs. The modifications required a minimum of three to five lines of additional code. Performance was measured by the time to code and debug the modifications and by the number of errors.

METHOD

Participants

Thirty-six professional programmers from three different locations participated in this experiment. All were General Electric employees. The participants averaged 8.5 years of professional programming experience (S.D. = 7.1) and had used an average of 5 programming languages (S.D. = 2.1).

Independent Variables

The experiment was designed to study the effects of three independent variables: the type of symbology, the spatial arrangement of the information and the type of program.

Program type. In our previous research (Sheppard, Curtis, Milliman & Love, 1979) significant differences in programmer performance were often associated with differences among programs. Three programs of varying types were chosen for use in this experiment. (These three programs were used in the first three experiments as well.) A program which calculated the trajectory of a rocket was chosen as representative of an engineering algorithm. An inventory system for a grocery distribution center represented the class of programs that

manipulate data bases. A third program combined these two types of applications. This program interrogated a data base for information concerning the traffic pattern at an airport and simulated future needs using a queueing algorithm.

These three programs were based on algorithms contained in Barrodale, Roberts, and Ehle (1971). The algorithms were modified to incorporate only the constructs of sequence, structured iteration, and structured selection. They were then coded in FORTRAN and verified for correctness. Each of the resulting programs contained approximately 50 lines of executable code. In addition a short algorithm (11 lines) was used as a practice program.

One modification was selected for each of the experimental programs. Prototype modifications were made to determine the minimum number of additional lines to complete the selected modifications. The rocket and inventory programs each required a minimum of three additional lines of code; the airport program required a minimum of five additional lines of code. Descriptions of the modifications and listings of the program code are presented in Appendix A.

Type of Symbology. The statements from each program were translated into three types of symbology: normal English, abbreviated English and a program design language (PDL).

Spatial Arrangements. Three spatial arrangements were used to represent the program structure: sequential, branching, and hierarchical. These three arrangements differed in the representation of control flow and nesting levels. In the sequential arrangement, both the control flow and the levels of nesting were represented vertically. In the branching arrangement, the flow of control was represented vertically while nesting levels were represented horizontally. Finally, in the hierarchical arrangement, the flow of control was represented horizontally while nesting levels were represented vertically.

Each of the three types of symbology was presented in the three spatial arrangements, resulting in nine documentation formats for each program. Appendix B of this report contains the nine formats for the rocket program.

Procedure

Prior to the experiment, the participants were given a 20-minute training session in which they were shown each spatial arrangement and each type of symbology. The experimenter described the control flow for each arrangement using a short program as an example; this program was not seen in the actual experiment. The procedure for using the text editor to modify the programs was also explained in detail during the training session.

Experimental sessions were conducted at CRT terminals on a VAX 11/780. All coding was done in FORTRAN. The participants were first given a practice program and a short description of the modification. The existing code could be listed on the CRT screen by using the editor. When satisfied that the modification was correct, a participant exited from the editor and activated a command file to compile and run the program. If the compilation was unsuccessful, a compiler message appeared on the screen directly below the line or lines containing the error. If the program compiled without errors, it was automatically executed with test data, and the output from the program appeared on the screen with one of the following messages: "OUTPUT IS CORRECT" or "OUTPUT IS INCORRECT." In the latter case, the participant was asked to keep trying until the program had been modified correctly.

Following the practice program, the three experimental programs were presented. For each program, the participants received a one-paragraph description of the modification. They also received a version of the documentation for the original (unmodified) program. The original code could be listed on the CRT screen. Finally they received a data dictionary listing each variable, a natural language description of it, and its data type.

The participants were told to make handwritten modifications on the documentation sheets before entering their code at the terminal. If a participant tried running the program

without making any changes, the program compiled successfully but produced the message that the output was incorrect.

An interactive data collection system prompted the participant throughout the experimental procedure. The system recorded each change made to a program. An interval timer, accurate to the nearest second, recorded the time for each action. When a participant required more than one editing session to modify the program and correct the errors, the experimental system recorded exits from the editor, any compilation errors, and the incorrect outputs generated. From these data, the time to modify the programs was calculated by summing the times from the individual editing sessions; time for compiling and running the programs was not included.

On the average, the participants spent approximately 27 minutes on each experimental program. They were required to continue working on a program until the modification had been completely successful. They were allowed to take breaks between programs.

Following the experiment, the participants completed a questionnaire about their previous programming experience. The information requested included number of years of professional experience, number of programming languages known, and whether they had previously worked with algorithms of the types used in the experiment. The participants were also asked about their preferences for type of symbology and spatial arrangement.

Design

The three types of symbology (normal English, abbreviated English, and PDL) were factorially combined with the three spatial arrangements (sequential, branching, and hierarchical) to produce nine documentation formats. These nine formats were constructed for each of the three programs, resulting in a total of 27 conditions.

Participants received a documentation format for each program. Across the three programs, they saw each type of symbology and each spatial arrangement. The first participant, for example, saw the rocket trajectory program presented in sequential normal English, the inventory control program in hierarchical PDL, and the airport traffic program in branching abbreviated English. The participants were assigned to conditions according to the procedures outlined in Winer (1971). [See also Kirk (1968)]. Each of the 27 conditions was used once within a set of nine participants. For this 3^3 randomized block design, a minimum of 36 participants is required to assess all interactions and main effects. Across the 36 participants, each program, symbology, and arrangement was presented first, second, and third an equal number of times.

RESULTS

Time to Modify and Debug

The participants required an average of 27 minutes to modify and debug a program. This represents the amount of time spent studying the program, modifying the documentation format and using the text editor (i.e., the total time spent at the terminal less the time for compiling, linking and running).

There were large differences in the times required to complete the modifications for the three programs (Table 1). The inventory program required the least time to complete (21.2 minutes); the airport program required the longest time (33.6 minutes).

TABLE 1. A COMPARISON OF THE DEPENDENT VARIABLES FOR THE THREE PROGRAMS

	PROGRAM			
	INVENTORY	ROCKET	AIRPORT	ALL PROGRAMS
MEAN TIME TO COMPLETE MODIFICATION (MINUTES)	21.2	24.9	33.6	26.6
MEAN NUMBER OF SEMANTIC ERRORS	0.8	1.2	1.2	1.1

The differences among the programs was verified by an analysis of variance ($p < .001$). (See Table 2.) A stepwise multiple regression equation was used to partition the sums of squares for the ANOVA. A logarithmic transformation was carried out on the times to attenuate the influence of extreme scores and to produce a more normal distribution (Kirk, 1968).

TABLE 2. SUMMARY OF ANOVA: TIME TO COMPLETE MODIFICATION

SOURCE	df	SS	MS	F	p
TOTAL	107	5.68			
BETWEEN PARTICIPANTS AND REPLICATIONS					
REPLICATIONS	3	.03			
PARTICIPANTS WITHIN REPLICATIONS	32	1.99			
WITHIN PARTICIPANTS AND REPLICATIONS					
PROGRAM (P)	2	.75	.38	12.7	.001
SYMBOLOGY (S)	2	.06	.03	1.0	
ARRANGEMENT (A)	2	.23	.12	4.0	.05
P × S	4	.34	.08	2.7	
P × A	4	.28	.07	2.3	
S × A	4	.15	.04	1.3	
P × S × A	8	.54	.07	2.3	
RESIDUAL	46	1.31	.03		

Table 3 presents the mean time to complete the modification for each combination of symbology and spatial arrangement. Differences due to the type of symbology were small. The PDL versions were associated with the smallest performance times for each spatial arrangement, but these differences were not statistically significant.

TABLE 3. MEAN TIME TO COMPLETE MODIFICATION (Minutes)

SPATIAL ARRANGEMENT	TYPE OF SYMBOLOGY			TOTAL
	NORMAL ENGLISH	ABBREVIATED ENGLISH	PROGRAM DESIGN LANGUAGE	
SEQUENTIAL	28.0	28.4	25.5	27.3
BRANCHING	25.4	22.9	21.1	23.1
HIERARCHICAL	30.9	28.6	28.3	29.3
TOTAL	28.1	26.6	25.0	26.6

Note: Individual cell means represent 12 participants.

A significant effect for spatial arrangement occurred ($p <$ .05). The branching versions required 23.1 minutes, while the sequential and hierarchical versions required 27.3 and 29.3 minutes respectively. There were no signficant two-way or three-way interactions.

Errors

The errors made by the participants provide insight on the difficulties encountered in making the modifications. Programs that did not compile and run successfully the first time were analyzed to determine what errors were present in the initial attempt to make the modification.

The errors were assigned to two general categories: syntactic and semantic. The syntactic category included a variety of errors that produced compiler messages. These errors were relatively few in number and were easy to detect and correct. Unlike the semantic errors, the syntactic errors could be corrected without reference to the instructions for the modification or to the documentation. Thus, they are of less interest than the semantic errors.

Table 1 shows a breakdown of the number of semantic errors for each program. The inventory program had fewer errors than the other two programs. A detailed analysis of the errors for the inventory program revealed that most of these errors (66%) resulted from problems in placing the statements in the correct locations within the program. The airport and rocket programs were associated with a wider variety of errors. Appendix C presents a detailed breakdown of the different types of errors for each program.

In terms of the symbology and spatial arrangement, the pattern of errors was similar to the pattern for the modification times. The effects of symbology were not pronounced, and the branching spatial arrangement was superior to the sequential and hierarchical arrangements (Table 4).

TABLE 4. MEAN NUMBER OF SEMANTIC ERRORS

SPATIAL ARRANGEMENT	TYPE OF SYMBOLOGY			TOTAL
	NORMAL ENGLISH	ABBREVIATED ENGLISH	PROGRAM DESIGN LANGUAGE	
SEQUENTIAL	1.3	1.8	0.8	1.3
BRANCHING	0.5	0.4	1.1	0.7
HIERARCHICAL	0.8	1.6	1.2	1.2
TOTAL	0.9	1.3	1.0	1.1

Preferences for Type of Symbology and Spatial Arrangement

Across the three programs, each participant received documentation in each type of symbology and in each spatial arrangement. The questionnaire indicated which three of the nine versions they had experienced during the experiment. They were asked to state which of these three versions they preferred. Table 5 shows these preferences.

TABLE 5. PERCENT OF PREFERENCES FOR SYMBOLOGY AND SPATIAL ARRANGEMENT

FACTOR	%
TYPE OF SYMBOLOGY:	
NORMAL ENGLISH	18
ABBREVIATED ENGLISH	32
PROGRAM DESIGN LANGUAGE	50
SPATIAL ARRANGEMENT:	
SEQUENTIAL	24
HIERARCHICAL	26
BRANCHING	50

In terms of the type of symbology, the majority of participants chose the constrained language, abbreviated English was intermediate, and natural English was the least preferred. The branching spatial arrangement was preferred twice as often as the sequential and hierarchical arrangements.

Experiential Factors as Predictors of Performance

Three factors relating to programming experience were compared to the participants' average performance on the experimental tasks. First, the practice (pretest) program was a common task done by all participants and could therefore be used as a measure of individual performance. The questionnaire provided information about two other experiential factors, the number of years of programming experience and the number of

programming languages used by the participants. Table 6 shows that the time spent on the pretest program was a predictor of performance time on the experimental programs ($r=.41$). The number of languages used by the participants was correlated with performance on both the pretest program (-.39) and the experimental programs (-.37). Finally, the number of years of programming experience did not show a significant correlation with any of the other measures.

TABLE 6. CORRELATIONS BETWEEN PERFORMANCE MEASURES AND EXPERIENTIAL FACTORS

	YEARS EXPERIENCE PROGRAMMING	NUMBER OF LANGUAGES	PRETEST TIME
MEAN TIME TO COMPLETE MODIFICATIONS	−.01	−.37*	.41**
PRETEST TIME	−.01	−.39**	
NUMBER OF LANGUAGES	.12		

$\underline{n} = 36$

*$p < .02$

**$p < .01$

DISCUSSION

Strong differences were observed among the three programs used in this experiment. The inventory control program was associated with the shortest times and fewest errors, the airport scheduling program resulted in the poorest performance, and the rocket trajectory program was in-between. This result parallels our past experiences with these programs in the comprehension and coding experiments. One explanation for the consistency of these results across several different programming tasks is that some types of algorithms are easier to understand and use than others. When asked whether they had previously worked with these types of algorithms, more participants said they had worked with an inventory control program (36%) than with rocket trajectory (19%) or airport scheduling programs (11%). Thus familiarity may account in part for performance differences among the programs.

Although the effect of type of symbology is not pronounced in this experiment, the results reflect the trend that appeared quite strongly in the previous three experiments. The more succinct symbology, the PDL, was associated with better performance than the more verbose symbology, the normal English. Further, the novel symbology introduced in the present experiment, the abbreviated English, was less verbose than the normal English but more verbose than the PDL. Performance times for the abbreviated English fell between the times for the normal English and the PDL, thus reinforcing the

conclusion that the more succinct the symbology, the more quickly the programming task will be completed.

The effect of spatial arrangement was quite strong in this experiment. The branching spatial arrangement was considerably better for the modification task than the other two arrangements. Similar evidence was obtained in the coding and comprehension experiments. In particular the branching spatial arrangement seems to be helpful in tasks related to the control flow structures of a program. In the coding experiment, fewer logical errors were associated with the branching arrangement than with the other two arrangements. In the comprehension experiment, the branching arrangement was superior for questions that required hand tracing through the program logic.

The participants' preferences for type of symbology and spatial arrangement in this experiment are consistent with preferences from the other experiments. PDL was the preferred symbology in this experiment as in the comprehension and coding experiments. (No preference for type of symbology was obtained in the debugging experiment.) The branching spatial arrangement was preferred in all four experiments.

As in our previous experiments, we compared performance to several experiential factors. Performance on the practice program was correlated with average performance on the experimental programs and was thus a good predictor of performance. Number of years of programming experience was not correlated with performance but number of programming languages

known was correlated with performance. Thus, diversity of experience is a better predictor of performance than length of experience. This replicates a similar result in our previous research (Sheppard et al., 1979) and highlights the importance of ensuring that programmers have an opportunity to gain broad applications experience as part of their professional development.

The four experiments in this series each produced slightly different results, depending on the four types of experimental tasks: answering questions, coding, debugging or modifying programs. No one particular combination of symbology and spatial arrangement proved superior for all tasks. However, one symbology, PDL, was associated with the best performance overall and was preferred most often by the participants.

Choice of spatial arrangement was not as clear. The sequential PDL was an excellent version. The hierarchical ideograms were also suprisingly usable in view of the participants' previous lack of experience with hierarchical versions of documentation. Overall, however, the branching spatial arrangement appeared to be associated with lower performance times and fewer errors than the other arrangements. Further, the branching arrangement was preferred in all four experiments. Software managers would be well advised to convert software specifications to PDL and should not feel constrained to the standard sequential format.

ACKNOWLEDGEMENTS

The authors would like to thank Joan Carter of GE's Computer Management Operations in Bridgeport, Connecticut and O.J. Barber and Roy Baessler of GE's Industrial Control Division in Charlottesville, Virginia for providing participants and facilities; Dave Morris and Pete McEvoy for designing the automatic data collection system; Dr. John O'Hare for advice, and Tom McDonald for preparing materials and statistical analyses.

REFERENCES

Barrodale, I., Roberts, F.D.K., & Ehle, B.L. Elementary computer applications in science, engineering, and business. New York: Wiley, 1971.

Jones, C. A survey of programming design and specification techniques. In Proceedings of the IEEE Conference on Specifications of Reliable Software. New York: Institute of Electrical and Electronics Engineers, 1979.

Kirk, R.E. Experimental design procedures for the behaviorial sciences. Belmont, CA: Brooks-Cole, 1968.

Ramsey, H.R., Atwood, M.E., & Van Doren, J.R. A comparative study of flowcharts and program design languages for the detailed procedural specification of computer programs. (Tech. Rep. #SAI-78-078-DEN). Denver: Science Applications, Inc. 1978.

Sheppard, S.B., Bailey, J.W., & Kruesi, E. The effects of the symbology and spatial arrangement of software specifications in a debugging task (Tech. Rep. TR-81-388200-4). Arlington, VA: General Electric, Information Systems Programs, 1981.

Sheppard, S.B., Curtis, B., Milliman, P., & Love, T. Modern coding practices and programmer performance. Computer, 1979, 12, (12), 41-49.

Sheppard, S.B. & Kruesi, E. The effects of the symbology and spatial arrangement of software specifications in a coding task. In Proceedings of Trends & Applications 1981: Advances in Software Technology, IEEE, 1981.

Sheppard, S.B., Kruesi, E., & Curtis, B. The effects of symbology and spatial arrangement on the comprehension of software specifications. In Proceedings of the Fifth International Conference on Software Engineering, IEEE, 1981.

Winer, B.J. Statistical principles in experimental design. New York: McGraw-Hill, 1971.

APPENDIX A

MODIFICATION DESCRIPTIONS AND PROGRAM LISTINGS

ROCKET PROBLEM

Program ROCKET currently assumes that the maximum time for the simulation, MAXT, is always less than the time for the total trajectory. The simulation always ends while the rocket is still airborne. More specifically, the simulation ends because MAXT has been exceeded and the variable FLAG has been changed. The flight director wants program ROCKET modified to include the option to stop the simulation when the rocket hits the ground. He would also like a message telling him which situation has occurred. If the rocket is airborne at the end of the simulation, the program should print the message: "ROCKET STILL ALOFT" and give the time. If the simulation ends because the rocket is no longer airborne, the program should print "ROCKET HIT GROUND" and give the time. (HINT: The rocket has hit the ground when the vertical distance, VDIST, is less than or equal to zero.) The message should be printed before the values for MASS, VACCEL, ..., and HDIST are printed. Formats 2000 and 3000 are included for your convenience.

PLEASE MAKE YOUR MODIFICATIONS ON THE SPECIFICATION SHEET BEFORE PROCEEDING TO CHANGE THE CODE.

INVENTORY PROBLEM

Program INVENTORY prints a separate invoice for each grocery store. Along with other information, the invoice lists each item ordered, the price per item and the total cost for that item. The manager of the chain of stores would like to have program INVENTORY modified to print a grand total at the end of each invoice. Use the variable name GTOTAL for the grand total. Format 150 is provided for your convenience.

PLEASE MAKE YOUR MODIFICATIONS ON THE SPECIFICATION SHEET BEFORE PROCEEDING TO CHANGE THE CODE.

AIRPORT PROBLEM

Assume that the FAA has imposed a new regulation concerning the amount of time an arriving airplane may remain in the air while waiting for a runway. If other runways are available but not being used, the longest time a pilot should wait is 5 minutes. Modify program AIRPORT to determine whether the maximum waiting time during simulation, MAXWT, has exceeded 5 minutes. You should also determine the value of a new variable, MAXARR, the maximum number of planes in ARRQUE, the arrival queue, at any time during the simulation. If MAXWT exceeds 5 minutes, print the message: "OPEN ANOTHER RUNWAY." Otherwise end the simulation with the message: "ANOTHER RUNWAY NOT NEEDED." In either case, print the value of MAXARR following the message and before the values for ENDT, ARRQUE, ..., NUMDEP. Formats 110, 120 and 130 are included for your convenience.

PLEASE MAKE YOUR MODIFICATIONS ON THE SPECIFICATION SHEET BEFORE PROCEEDING TO CHANGE THE CODE.

ROCKET PROGRAM

```
C       PROGRAM ROCKET
        INTEGER MAXT, TIME, FLAG
        REAL  VACCEL, VVELOC, VDIST, HACCEL, HVELOC, HDIST,
     1  ANGLE, TILT, GRAV, MASS, FUEL, FORCE
        OPEN  (UNIT=1, NAME='MAX. DAT', TYPE='OLD')
        OPEN  (UNIT=3, NAME='RUN. DAT', TYPE='NEW')
        OPEN(UNIT=4, NAME='TERM. DAT', TYPE='NEW')
  3001  FORMAT(1H1)
        WRITE(6, 3001)
C
C
C    INITIALIZATION
C
C
        VACCEL = 0.
        VVELOC = 0.
        VDIST = 0.
        HACCEL = 0.
        HVELOC = 0.
        HDIST = 0.
        ANGLE = 0.
        TILT = 0. 3491
        GRAV = 32.
        MASS = 10000.
        FUEL = 50.
        FORCE = 400000.
        READ(1, 1000) MAXT
        FLAG = 0
        TIME = 1
C
C
C    COMPUTATION:
C
C
   10   IF (FLAG .NE.  0) GO TO 60
        IF (TIME .GT.  100) GO TO 20
        MASS = MASS - FUEL
        IF (TIME .NE.  11) GO TO 30
        ANGLE = TILT
        GO TO 30
   20   IF (TIME .NE.  101) GO TO 30
        FORCE = 0. 0
   30   VACCEL = ((FORCE * COS(ANGLE))/MASS) - GRAV
        VVELOC = VVELOC + VACCEL
        VDIST = VDIST + VVELOC
        HACCEL = (FORCE * SIN(ANGLE))/MASS
        HVELOC = HVELOC + HACCEL
        HDIST = HDIST + HVELOC
        TIME = TIME + 1
        IF (TIME .GT.  MAXT) FLAG =1      IF (VDIST .LE. Ø) FLAG = 1
        GO TO 10
```

```
C
C
C     TERMINATION:
C
C                                          IF (VDIST.GT.Ø) WRITE (4,2ØØØ) TIME
   60    TIME = TIME - 1                   IF (VDIST.LE.Ø) WRITE (4,3ØØØ) TIME
         WRITE(4,4000)  MASS, VACCEL, VVELOC, VDIST,
     1   HACCEL,HVELOC,HDIST
         CLOSE(UNIT=4)
         CALL CHECK4(1,TIME,0,0,0,0,0,0,0)




         WRITE(3,4000)MASS,VACCEL,VVELOC,VDIST,
     1   HACCEL,HVELOC,HDIST
         WRITE(6,4000)MASS,VACCEL,VVELOC,VDIST,
     1   HACCEL,HVELOC,HDIST
         CLOSE(UNIT=3)
         CLOSE(UNIT=1)
         STOP
 1000    FORMAT(I3)
 2000    FORMAT(5X,'ROCKET STILL ALOFT AT TIME =',
     1   I5,' SECONDS')
 3000    FORMAT(5X,'ROCKET HIT GROUND AT TIME=  ',I5,' SECONDS')
 4000    FORMAT(5X,'MASS = ',F22.2/
     1   5X,'VERTICAL ACCEL = ',F12.2/
     2   5X,'VERTICAL VELOC = ',F12.2/
     3   5X,'VERTICAL DIST = ',F13.2/
     4   5X,'HORIZONTAL ACCEL = ',F10.2/
     5   5X,'HORIZONTAL VELOC = ',F10.2/
     6   5X,'HORIZONTAL DIST = ',F11.2)
         END
```

INVENTORY PROGRAM

```
C        PROGRAM INVENTORY
         INTEGER DELIV, FLAG, ITEM, ONHAND, ORDER, RELEV,
     1   REORD, STORE, UNFILL
         REAL  PRICE, TOTAL
C
         GTOTAL = -1.0
         WRITE(6,3001)
  3001   FORMAT(1H1)
C
C    INITIALIZATION:
C
C
         OPEN (UNIT=4, NAME='TERM.DAT', TYPE = 'NEW')
         OPEN (UNIT=1, NAME='ORDERS.DAT', TYPE='OLD')
         OPEN (UNIT=2, NAME='PURCHAS.DAT', TYPE='OLD',
     1   ACCESS='SEQUENTIAL')
         OPEN (UNIT=3,NAME='RUN.DAT',TYPE='NEW')
C
C
C    COMPUTATION:
C
C
         CALL SETUP
    10   READ (1, 100, END=80) STORE
         WRITE (4, 110) STORE              GTOTAL = 0.0
    20   READ (1, 120) ITEM, ORDER
         IF (ITEM .EQ. 0) GO TO 70
         CALL FETCH2(ITEM, PRICE, ONHAND, RELEV, REORD, FLAG)
         IF (ONHAND .LE. ORDER) GO TO 30
         DELIV = ORDER
         ONHAND = ONHAND - ORDER
         UNFILL = 0
         GO TO 40
    30   DELIV = ONHAND
         ONHAND = 0
         UNFILL = ORDER - DELIV
    40   IF (ONHAND .GT. RELEV) GO TO 50
         IF (FLAG .EQ. 0)  FLAG = 1
    50   TOTAL = DELIV * PRICE             GTOTAL = GTOTAL + TOTAL
         IF (FLAG .NE. 1) GO TO 60
         WRITE (2, 130) ITEM, REORD
         FLAG = 2
    60   WRITE(4,140) ITEM, PRICE, ORDER, DELIV, UNFILL,TOTAL
         CALL UPDATE (ITEM, ONHAND, FLAG)
         GO TO 20
    70   CONTINUE
         GO TO 10                          WRITE (4,150) GTOTAL
```

```
C
C
C     TERMINATION:
C
C
   80 CLOSE (UNIT=1)
      CLOSE(UNIT=4)
      CLOSE (UNIT=2)
      CALL CHECK4(2,0,0,0,0,0,0,GTOTAL,0)
      WRITE (3,140) ITEM, PRICE, ORDER, DELIV, UNFILL,TOTAL
      WRITE (3, 150) GTOTAL
      CLOSE (UNIT=3)
   90 CONTINUE
      STOP


  100 FORMAT (I2)
  110 FORMAT (//, 5X, 'INVOICE FOR STORE NUMBER: ', I3)
  120 FORMAT (I3, I5)
  130 FORMAT (2I7)
  140 FORMAT (5X, 'ITEM NUMBER: ', I11 / 5X,
     1 'PRICE PER ITEM:  $', F5.2 / 5X, 'NUMBER ORDERED: ',
     2 I8,/5X, 'NUMBER DELIVERED: ', I6/ 5X,
     3 'UNABLE TO DELIVER: ',I5/5X, 'TOTAL PRICE:  $', F8.2)
  150 FORMAT (/,5X, 'TOTAL PRICE FOR ALL ITEMS: $', F10.2)
      END
```

AIRPORT PROGRAM

```
C       PROGRAM  AIRPORT
        INTEGER  ARRQUE, BEGINT, CLEAR, DEPQUE, ENDT, MAXWT
        INTEGER  NUMARR, NUMDEP, TIME
        REAL  ARPROB, DPPROB, RAND1, RAND2, RSEED
        OPEN (UNIT=3,NAME='RUN.DAT',TYPE='NEW')
  3001  FORMAT(1H1)
        OPEN (UNIT=4,NAME='TERM.DAT',TYPE='NEW')
        MAXARR = 999
        WRITE(6,3001)
C
C
C    INITIALIZATION:
C
C       <---------- MAXARR = Ø
        CALL FETCH1(BEGINT, ARPROB, DPPROB, ARRQUE, DEPQUE,
     1  CLEAR)
        RSEED = 0.0
        NUMARR = 0
        NUMDEP = 0
        TIME = BEGINT
        ENDT = BEGINT + 20
C
C
C    COMPUTATION:
C
C
    10  IF (TIME .GT. ENDT) GO TO 60
        RAND1 = RND(RSEED)
        IF (RAND1 .GT. ARPROB) GO TO 20
        ARRQUE = ARRQUE + 1
    20  RAND2 = RND(RSEED) <---- IF (ARRQUE .GT. MAXARR) MAXARR = ARRQUE
        IF (RAND2 .GT. DPPROB) GO TO 30
        DEPQUE = DEPQUE + 1
    30  IF (CLEAR .GT. TIME) GO TO 50
        IF (ARRQUE .LE. 0) GO TO 40
        ARRQUE = ARRQUE - 1
        NUMARR = NUMARR + 1
        CLEAR = TIME + 3
        GO TO 50
    40  IF (DEPQUE .LE. 0) GO TO 50
        DEPQUE = DEPQUE - 1
        NUMDEP = NUMDEP + 1
        CLEAR = TIME + 2
    50  TIME = TIME + 1
        GO TO 10
    60  MAXWT = (CLEAR-ENDT) + (ARRQUE*3) + (DEPQUE*2)
```

```
C
C
C    TERMINATION:               IF (MAXWT .GT. 5) WRITE (4,110)
C                               IF (MAXWT .LE. 5) WRITE (4,120)
C                               WRITE (4,130) MAXARR
  80     WRITE (4, 100) ENDT, ARRQUE, NUMARR, DEPQUE,
     1   NUMDEP
         CLOSE(UNIT=4)
         CALL CHECK4(3,MAXARR,0,0,0,0,0,0,0)
         WRITE(6,100) ENDT,ARRQUE,NUMARR,DEPQUE,NUMDEP
         WRITE (3,100) ENDT,ARRQUE,NUMARR,DEPQUE,NUMDEP
         IF (MAXARR .NE. 999) WRITE (6,130) MAXARR
         WRITE (3,130) MAXARR
         CLOSE (UNIT=3)
         STOP



 100     FORMAT (6X, 'ENDING TIME FOR SIMULATION: ', I5/
     1   12X,'ARRIVAL QUEUE: ', I5/11X,'NUMBER ARRIVED: ',I5/
     1   10X,'DEPARTURE QUEUE: ', I5/10X,'NUMBER DEPARTED: ',
     1   I5)
 110     FORMAT (5X,  'OPEN ANOTHER RUNWAY          ')
 120     FORMAT (5X,  'ANOTHER RUNWAY NOT NEEDED    ')
 130     FORMAT (5X,'MAXIMUM # OF ARRIVALS IS ',I5)
         END
```

APPENDIX B

DOCUMENTATION FORMATS FOR ROCKET PROGRAM

PROGRAM TO SIMULATE THE PATH OF A ROCKET

SET THE FOLLOWING VARIABLES TO ZERO:
(A) THE VERTICAL ACCELERATION,
(B) THE VERTICAL VELOCITY,
(C) THE VERTICAL DISTANCE,
(D) THE HORIZONTAL ACCELERATION,
(E) THE HORIZONTAL VELOCITY,
(F) THE HORIZONTAL DISTANCE,
(G) THE ANGLE AT WHICH THE ROCKET IS AIMED.
SET THE DEGREE OF TILT TO 0.3491.
SET THE GRAVITATIONAL PULL TO 32.
SET THE TOTAL MASS OF THE ROCKET TO 10000.
SET THE AMOUNT OF FUEL EXPENDED PER SECOND TO 50.
SET THE FORCE OF THE ROCKET TO 400000.

READ THE MAXIMUM TIME FOR THE SIMULATION FROM THE FILE 'MAX'.

SET THE FLAG TO ZERO.
SET THE SIMULATION TIME TO ONE.

DO STEPS 1 THROUGH 4 WHILE THE FLAG IS ZERO.

1. IF THE SIMULATION TIME IS LESS THAN OR EQUAL TO 100, DECREASE THE MASS OF THE ROCKET BY THE AMOUNT OF FUEL EXPENDED PER SECOND AND CHECK TO SEE IF THE TIME EQUALS 11; IF SO, SET THE ANGLE OF THE ROCKET TO THE DEGREE OF TILT. OTHERWISE (IF THE SIMULATION TIME IS GREATER THAN 100) CHECK TO SEE IF IT EQUALS 101; IF SO, SET THE FORCE OF THE ROCKET TO ZERO.

1. IF THE SIMULATION TIME IS LESS THAN OR EQUAL TO 100, DECREASE THE MASS OF THE ROCKET BY THE AMOUNT OF FUEL EXPENDED PER SECOND AND CHECK TO SEE IF THE TIME EQUALS 11; IF SO, SET THE ANGLE OF THE ROCKET TO THE DEGREE OF TILT. OTHERWISE (IF THE SIMULATION TIME IS GREATER THAN 100) CHECK TO SEE IF IT EQUALS 101; IF SO, SET THE FORCE OF THE ROCKET TO ZERO.

2. CALCULATE THE VERTICAL ACCELERATION AS FOLLOWS:
   MULTIPLY THE FORCE OF THE ROCKET BY THE COSINE OF THE ANGLE AT WHICH THE ROCKET IS AIMED; DIVIDE THIS QUANTITY BY THE MASS OF THE ROCKET AND THEN SUBTRACT THE GRAVITATIONAL PULL.
   INCREASE THE VERTICAL VELOCITY BY THE VERTICAL ACCELERATION.
   INCREASE THE VERTICAL DISTANCE BY THE VERTICAL VELOCITY.

   CALCULATE THE HORIZONTAL ACCELERATION AS FOLLOWS:
   MULTIPLY THE FORCE OF THE ROCKET BY THE SINE OF THE ANGLE AT WHICH THE ROCKET IS AIMED AND DIVIDE THIS QUANTITY BY THE MASS OF THE ROCKET.
   INCREASE THE HORIZONTAL VELOCITY BY THE HORIZONTAL ACCELERATION.
   INCREASE THE HORIZONTAL DISTANCE BY THE HORIZONTAL VELOCITY.

3. INCREASE THE SIMULATION TIME BY ONE.

4. IF THE SIMULATION TIME IS GREATER THAN THE MAXIMUM TIME, SET THE FLAG TO ONE.

DECREASE THE SIMULATION TIME BY ONE.

PRINT THE MASS OF THE ROCKET, THE VERTICAL ACCELERATION, THE VERTICAL VELOCITY, THE VERTICAL DISTANCE, THE HORIZONTAL ACCELERATION, THE HORIZONTAL VELOCITY, AND THE HORIZONTAL DISTANCE.

THIS COMPLETES THE PROCESS NECESSARY TO SIMULATE THE ROCKET PATH.

NORMAL ENGLISH — BRANCHING

PROGRAM TO SIMULATE THE PATH OF A ROCKET

SET THE FOLLOWING VARIABLES TO ZERO:
(A) THE VERTICAL ACCELERATION,
(B) THE VERTICAL VELOCITY,
(C) THE VERTICAL DISTANCE,
(D) THE HORIZONTAL ACCELERATION,
(E) THE HORIZONTAL VELOCITY,
(F) THE HORIZONTAL DISTANCE,
(G) THE ANGLE AT WHICH THE ROCKET IS AIMED.
SET THE DEGREE OF TILT TO 0.3491.
SET THE GRAVITATIONAL PULL TO 32.
SET THE TOTAL MASS OF THE ROCKET TO 10000.
SET THE AMOUNT OF FUEL EXPENDED PER SECOND TO 50.
SET THE FORCE OF THE ROCKET TO 400000.

READ THE MAXIMUM TIME FOR THE SIMULATION FROM THE FILE 'MAX'.

SET THE FLAG TO ZERO.
SET THE SIMULATION TIME TO ONE.

DO THE STEPS TO THE RIGHT WHILE THE FLAG IS ZERO

IF THE SIMULATION TIME IS LESS THAN OR EQUAL TO 100

THEN

DECREASE THE MASS OF THE ROCKET BY THE AMOUNT OF FUEL EXPENDED PER SECOND.

IF THE TIME EQUALS 11

THEN

SET THE ANGLE OF THE ROCKET TO THE DEGREE OF TILT.

OTHERWISE

OTHERWISE

IF THE SIMULATION TIME EQUALS 101

THEN

SET THE FORCE OF THE ROCKET TO ZERO.

OTHERWISE

CALCULATE THE VERTICAL ACCELERATION AS FOLLOWS: MULTIPLY THE FORCE OF THE ROCKET BY THE COSINE OF THE ANGLE AT WHICH THE ROCKET IS AIMED; DIVIDE THIS QUANTITY BY THE MASS OF THE ROCKET AND THEN SUBTRACT THE GRAVITATIONAL PULL.

INCREASE THE VERTICAL VELOCITY BY THE VERTICAL

MULTIPLY THE FORCE OF THE ROCKET BY THE COSINE OF THE ANGLE AT WHICH THE ROCKET IS AIMED; DIVIDE THIS QUANTITY BY THE MASS OF THE ROCKET AND THEN SUBTRACT THE GRAVITATIONAL PULL.

INCREASE THE VERTICAL VELOCITY BY THE VERTICAL ACCELERATION.

INCREASE THE VERTICAL DISTANCE BY THE VERTICAL VELOCITY.

CALCULATE THE HORIZONTAL ACCELERATION AS FOLLOWS: MULTIPLY THE FORCE OF THE ROCKET BY THE SINE OF THE ANGLE AT WHICH THE ROCKET IS AIMED AND DIVIDE THIS QUANTITY BY THE MASS OF THE ROCKET.

INCREASE THE HORIZONTAL VELOCITY BY THE HORIZONTAL ACCELERATION.

INCREASE THE HORIZONTAL DISTANCE BY THE HORIZONTAL VELOCITY.

INCREASE THE SIMULATION TIME BY ONE.

IF THE SIMULATION TIME IS GREATER THAN THE MAXIMUM TIME

THEN

OTHERWISE

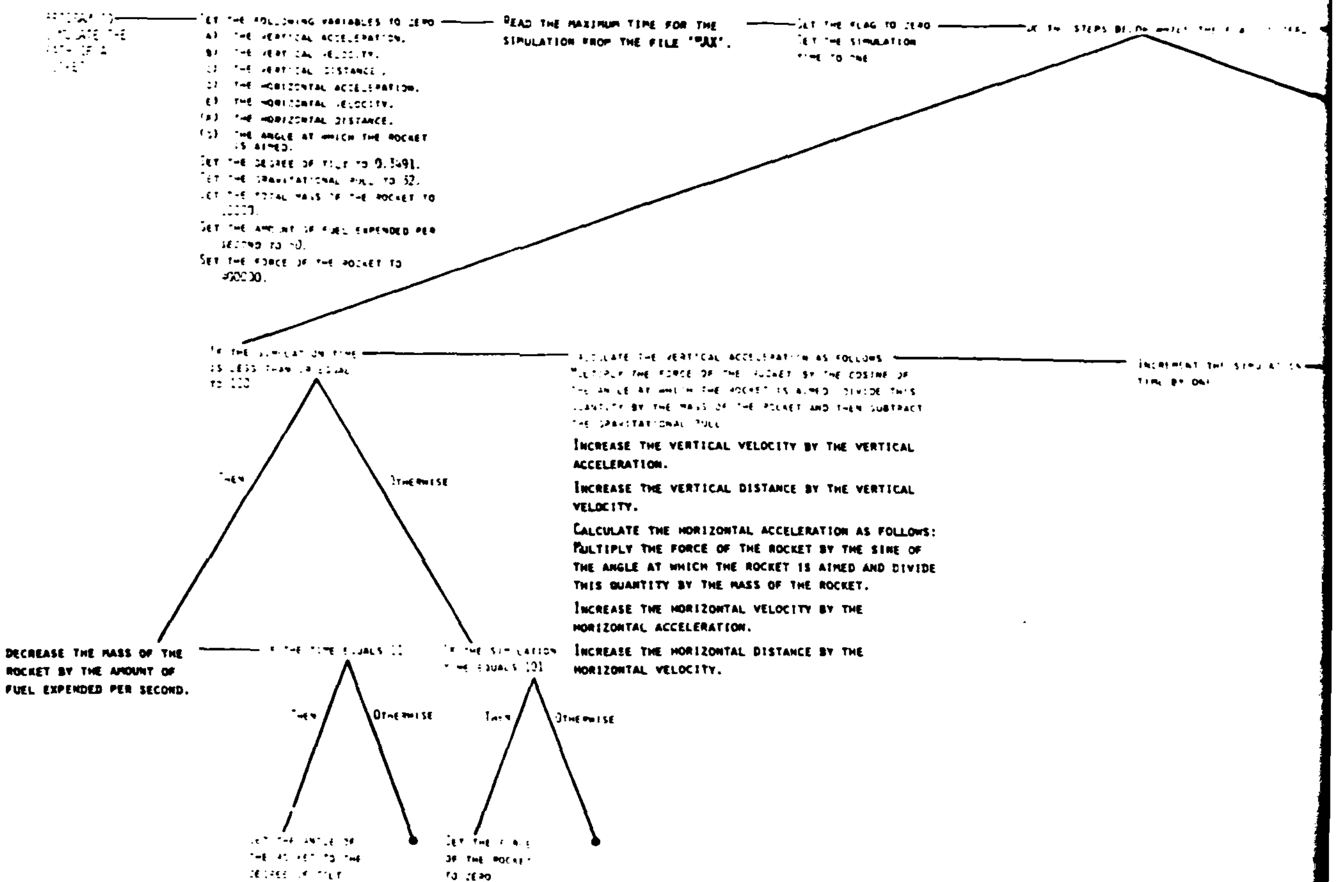
SET THE FLAG TO ONE.

DECREASE THE SIMULATION TIME BY ONE.

PRINT THE MASS OF THE ROCKET, THE VERTICAL ACCELERATION, THE VERTICAL VELOCITY, THE VERTICAL DISTANCE, THE HORIZONTAL ACCELERATION, THE HORIZONTAL VELOCITY, AND THE HORIZONTAL DISTANCE.

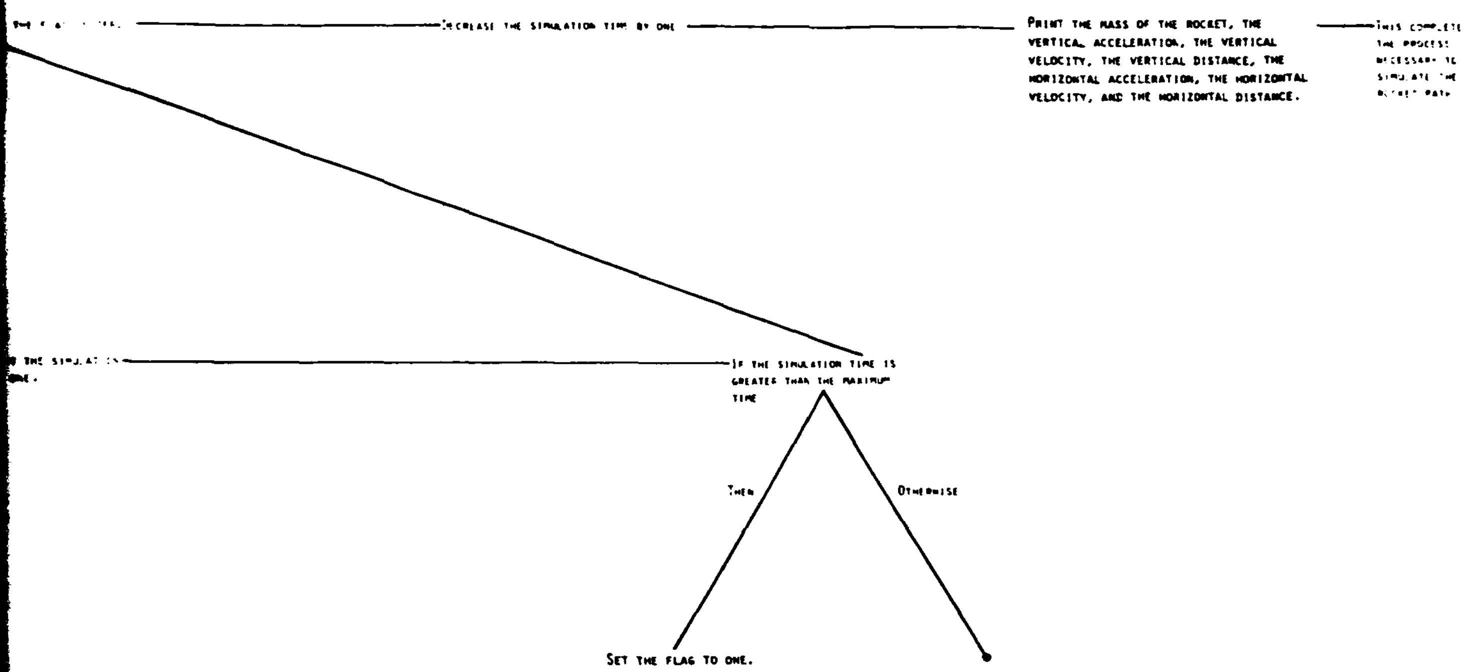
THIS COMPLETES THE PROCESS NECESSARY TO SIMULATE THE ROCKET PATH.

,2



SET THE FOLLOWING VARIABLES TO ZERO
A) THE VERTICAL ACCELERATION.
B) THE VERTICAL VELOCITY.
C) THE VERTICAL DISTANCE.
D) THE HORIZONTAL ACCELERATION.
E) THE HORIZONTAL VELOCITY.
(F) THE HORIZONTAL DISTANCE.
(G) THE ANGLE AT WHICH THE ROCKET IS AIMED.
SET THE DEGREE OF TILT TO 0.3491.
SET THE GRAVITATIONAL PULL TO 32.
SET THE TOTAL MASS OF THE ROCKET TO
SET THE AMOUNT OF FUEL EXPENDED PER SECOND
SET THE FORCE OF THE ROCKET TO
READ THE MAXIMUM TIME FOR THE SIMULATION FROM THE FILE "MAX".
SET THE FLAG TO ZERO
SET THE SIMULATION TIME TO ONE
IF THE SIMULATION TIME IS LESS THAN OR EQUAL TO
THEN
OTHERWISE
DECREASE THE MASS OF THE ROCKET BY THE AMOUNT OF FUEL EXPENDED PER SECOND.
IF THE SIMULATION TIME EQUALS 101
THEN
OTHERWISE
THEN
OTHERWISE
SET THE FORCE OF THE ROCKET TO ZERO
CALCULATE THE VERTICAL ACCELERATION AS FOLLOWS
MULTIPLY THE FORCE OF THE ROCKET BY THE COSINE OF THE ANGLE AT WHICH THE ROCKET IS AIMED. DIVIDE THIS QUANTITY BY THE MASS OF THE ROCKET AND THEN SUBTRACT THE GRAVITATIONAL PULL
INCREASE THE VERTICAL VELOCITY BY THE VERTICAL ACCELERATION.
INCREASE THE VERTICAL DISTANCE BY THE VERTICAL VELOCITY.
CALCULATE THE HORIZONTAL ACCELERATION AS FOLLOWS:
MULTIPLY THE FORCE OF THE ROCKET BY THE SINE OF THE ANGLE AT WHICH THE ROCKET IS AIMED AND DIVIDE THIS QUANTITY BY THE MASS OF THE ROCKET.
INCREASE THE HORIZONTAL VELOCITY BY THE HORIZONTAL ACCELERATION.
INCREASE THE HORIZONTAL DISTANCE BY THE HORIZONTAL VELOCITY.
TIME BY ONE

NORMAL ENGLISH — HIERARCHICAL



, 2

PROGRAM TO SIMULATE THE PATH OF A ROCKET

ABBREVIATED ENGLISH — SEQUENTIAL

SET THE FOLLOWING VARIABLES TO ZERO:

(A) VACCEL
(B) VVELOC
(C) VDIST
(D) HACCEL
(E) HVELOC
(F) HDIST
(G) ANGLE
SET TILT TO 0.3491.
SET GRAV TO 32.
SET MASS TO 10000.
SET FUEL TO 50.
SET FORCE TO 400000.

READ MAXT FROM THE FILE 'MAX'.

SET FLAG TO ZERO.

SET TIME TO ONE.

DO STEPS 1 THROUGH 4 WHILE FLAG IS ZERO.

1. IF TIME IS LESS THAN OR EQUAL TO 100, DECREASE MASS BY FUEL AND CHECK TO SEE IF TIME EQUALS 11; IF SO, SET ANGLE TO TILT.

   OTHERWISE (IF TIME IS GREATER THAN 100) CHECK TO SEE IF IT EQUALS 101; IF SO, SET FORCE TO ZERO.

2. CALCULATE VACCEL AS FOLLOWS:

1. IF TIME IS LESS THAN OR EQUAL TO 100, DECREASE MASS BY FUEL AND CHECK TO SEE IF TIME EQUALS 11; IF SO, SET ANGLE TO TILT.

   OTHERWISE (IF TIME IS GREATER THAN 100) CHECK TO SEE IF IT EQUALS 101; IF SO, SET FORCE TO ZERO.

2. CALCULATE VACCEL AS FOLLOWS:
   MULTIPLY FORCE BY THE COSINE OF ANGLE; DIVIDE THIS QUANTITY BY MASS AND THEN SUBTRACT GRAV.
   INCREASE VVELOC BY VACCEL.
   INCREASE VDIST BY VVELOC.
   CALCULATE HACCEL AS FOLLOWS:
   MULTIPLY FORCE BY THE SINE OF ANGLE AND DIVIDE THIS QUANTITY BY MASS.
   INCREASE HVELOC BY HACCEL.
   INCREASE HDIST BY HVELOC.

3. INCREASE TIME BY ONE.

4. IF TIME IS GREATER THAN MAXT, SET FLAG TO ONE.

DECREASE TIME BY ONE.

PRINT MASS, VACCEL, VVELOC, VDIST, HACCEL, HVELOC, AND HDIST.

THIS COMPLETES THE PROCESS NECESSARY TO SIMULATE THE ROCKET PATH.

. 2

ABBREVIATED ENGLISH — BRANCHING

PROGRAM TO SIMULATE THE PATH
OF A ROCKET

SET THE FOLLOWING VARIABLES TO ZERO:
(A) VACCEL
(B) VVELOC
(C) VDIST
(D) HACCEL
(E) HVELOC
(F) HDIST
(G) ANGLE
SET TILT TO 0.3491.
SET GRAV TO 32.
SET MASS TO 10000.
SET FUEL TO 50.
SET FORCE TO 400000.

READ MAXT FROM THE FILE 'MAX'.

SET FLAG TO ZERO.
SET TIME TO ONE.

DO THE STEPS TO THE RIGHT WHILE FLAG IS ZERO.

IF TIME IS LESS THAN
OR EQUAL TO 100

THEN

DECREASE MASS BY FUEL

IF TIME EQUALS 11

THEN

SET ANGLE TO TILT.

OTHERWISE

OTHERWISE

IF TIME
EQUALS 101

THEN

SET FORCE TO ZERO.

OTHERWISE

CALCULATE VACCEL AS FOLLOWS:
MULTIPLY FORCE BY THE COSINE OF ANGLE;
DIVIDE THIS QUANTITY BY
MASS AND THEN SUBTRACT GRAV.
INCREASE VVELOC BY VACCEL.

Multiply FORCE by the cosine of ANGLE;
divide this quantity by
MASS and then subtract GRAV.
Increase VVELOC by VACCEL.
Increase VDIST by VVELOC.
Calculate HACCEL as follows:
Multiply FORCE by the sine of ANGLE
and divide this quantity by MASS.
Increase HVELOC by HACCEL.
Increase HDIST by HVELOC.

Increase TIME by one.

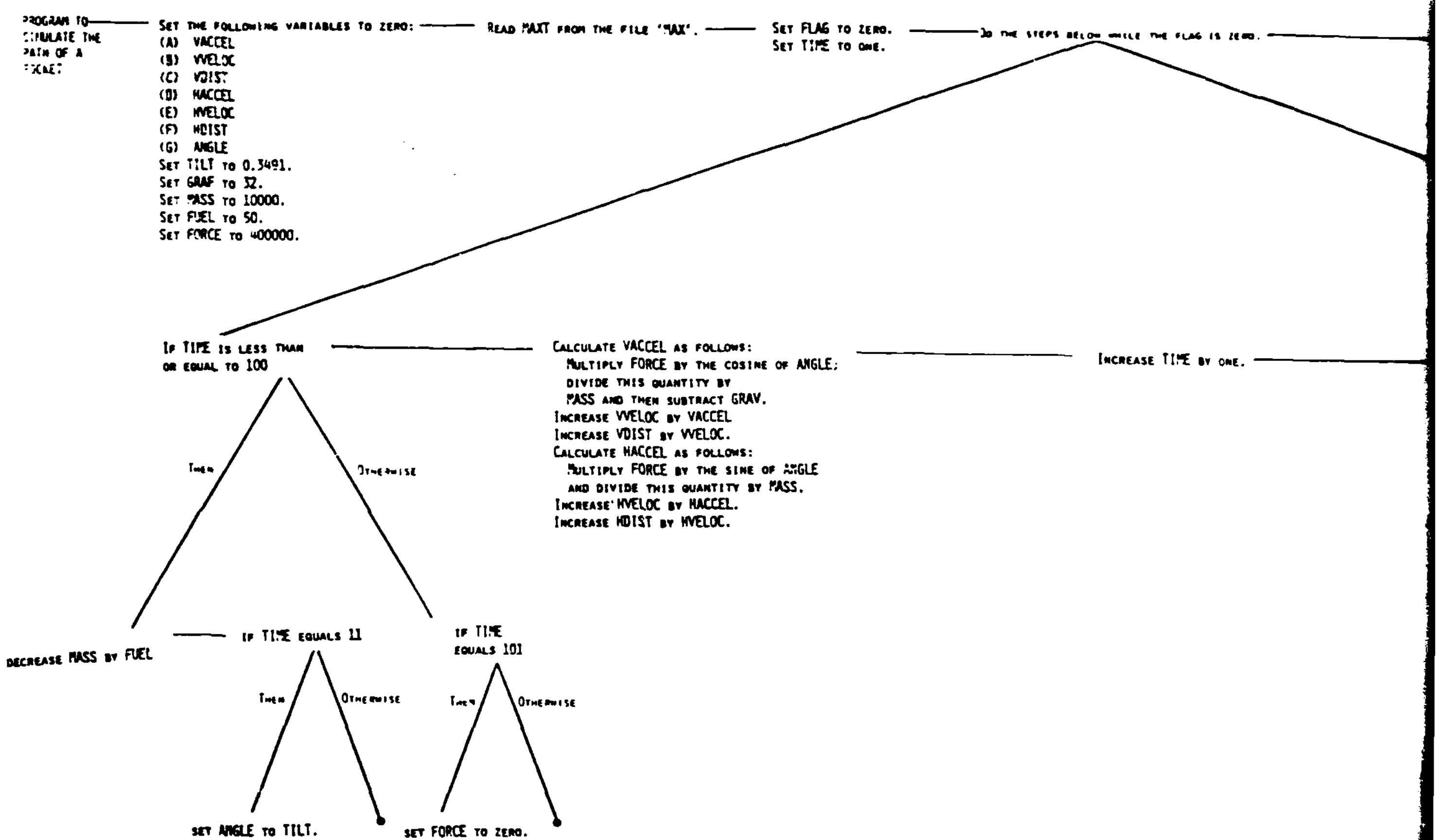
If TIME is greater than MAXT

Then	Otherwise
Set FLAG to one.

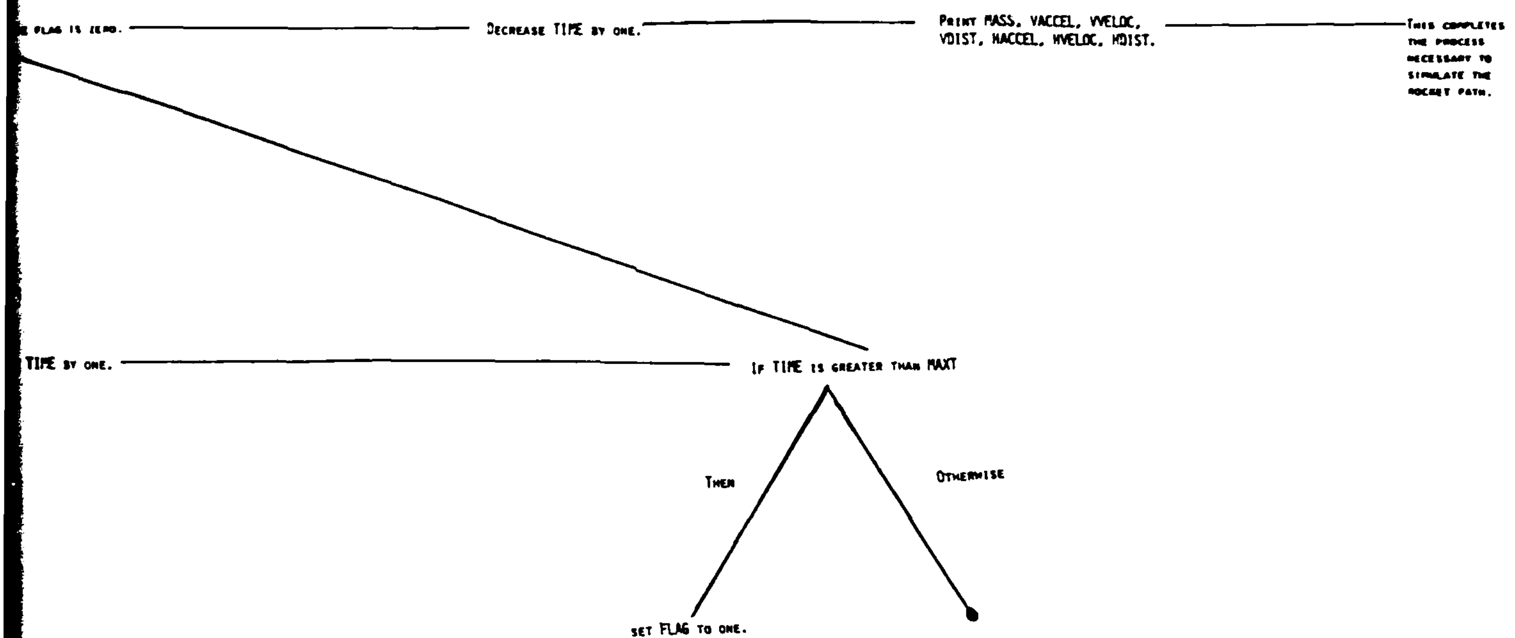
Decrease TIME by one.

Print MASS, VACCEL, VVELOC,
VDIST, HACCEL, HVELOC, HDIST.

This completes the process
necessary to simulate the
rocket path.


PROGRAM TO SIMULATE THE PATH OF A ROCKET
SET THE FOLLOWING VARIABLES TO ZERO:
(A) VACCEL
(B) VVELOC
(C) VDIST
(D) HACCEL
(E) HVELOC
(F) HDIST
(G) ANGLE
SET TILT TO 0.3491.
SET GRAF TO 32.
SET MASS TO 10000.
SET FUEL TO 50.
SET FORCE TO 400000.
READ MAXT FROM THE FILE 'MAX'.
SET FLAG TO ZERO.
SET TIME TO ONE.
DO THE STEPS BELOW WHILE THE FLAG IS ZERO.
IF TIME IS LESS THAN OR EQUAL TO 100
THEN
OTHERWISE
DECREASE MASS BY FUEL
IF TIME EQUALS 11
THEN
OTHERWISE
SET ANGLE TO TILT.
IF TIME EQUALS 101
THEN
OTHERWISE
SET FORCE TO ZERO.
CALCULATE VACCEL AS FOLLOWS:
MULTIPLY FORCE BY THE COSINE OF ANGLE;
DIVIDE THIS QUANTITY BY
MASS AND THEN SUBTRACT GRAV.
INCREASE VVELOC BY VACCEL
INCREASE VDIST BY VVELOC.
CALCULATE HACCEL AS FOLLOWS:
MULTIPLY FORCE BY THE SINE OF ANGLE
AND DIVIDE THIS QUANTITY BY MASS.
INCREASE HVELOC BY HACCEL.
INCREASE HDIST BY HVELOC.
INCREASE TIME BY ONE.

ABBREVIATED ENGLISH — HIERARCHICAL



2

PDL — SEQUENTIAL

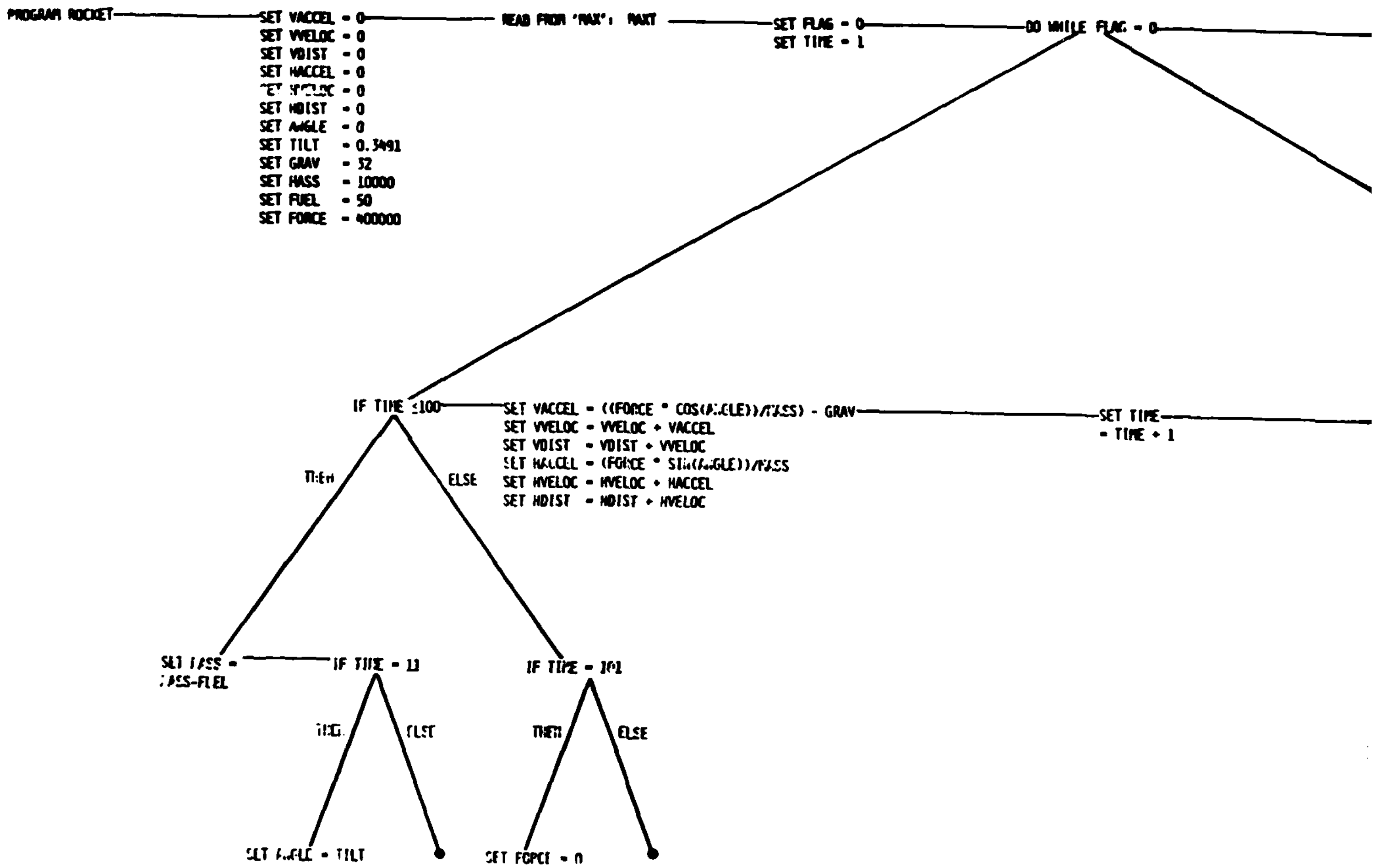
```
PROGRAM ROCKET

SET VACCEL = 0
SET VVELOC = 0
SET VDIST  = 0
SET HACCEL = 0
SET HVELOC = 0
SET HDIST  = 0
SET ANGLE  = 0
SET TILT   = 0.3491
SET GRAV   = 32
SET MASS   = 10000
SET FUEL   = 50
SET FORCE  = 400000

READ FROM 'MAX':  MAXT

SET FLAG   = 0
SET TIME   = 1

DO WHILE FLAG = 0

     IF TIME ≤ 100

          THEN

          SET MASS = MASS-FUEL

          IF TIME = 11

               THEN

               SET ANGLE = TILT

          ENDIF

          ELSE

          IF TIME = 101

               THEN

               SET FORCE =

          ENDIF

     ENDIF
```

```
                THEN

                SET FORCE = 0

            ENDIF

        ENDIF

        SET VACCEL = ((FORCE * COS(ANGLE))/MASS) - GRAV
        SET VVELOC =  VVELOC + VACCEL
        SET VDIST  =  VDIST + VVELOC
        SET HACCEL = (FORCE * SIN(ANGLE))/MASS
        SET HVELOC =  HVELOC + HACCEL
        SET HDIST  =  HDIST + HVELOC

        SET TIME = TIME + 1

        IF TIME > MAXT

            THEN

            SET FLAG = 1

        ENDIF

    ENDDO

    SET TIME = TIME - 1

    PRINT, MASS, VACCEL,
    VVELOC, VDIST, HACCEL,
    HVELOC, HDIST

    END OF ROCKET
```

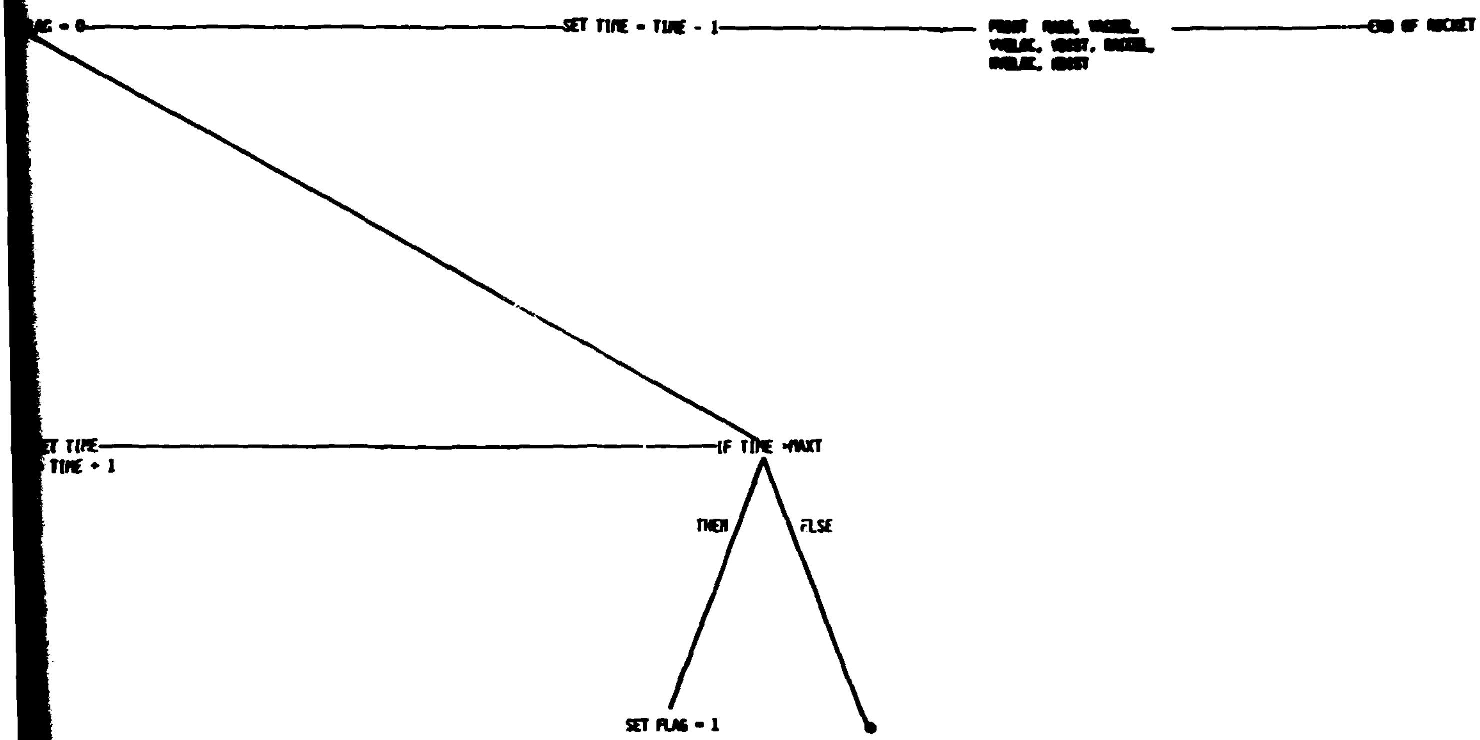
PDL — BRANCHING

```
PROGRAM ROCKET

SET VACCEL  = 0
SET VVELOC  = 0
SET VDIST   = 0
SET HACCEL  = 0
SET HVELOC  = 0
SET HDIST   = 0
SET ANGLE   = 0
SET TILT    = 0.3491
SET GRAV    = 32
SET MASS    = 10000
SET FUEL    = 50
SET FORCE   = 400000

READ FROM 'MAX', MAXT

SET FLAG = 0
SET TIME = 1

DO WHILE FLAG = 0
    IF TIME ≤ 100
    THEN
        SET MASS = MASS-FUEL
        IF TIME = 11
        THEN
            SET ANGLE = TILT
        ELSE
    ELSE
        IF TIME = 101
        THEN
            SET FORCE = 0
        ELSE

    SET VACCEL = ((FORCE * COS(ANGLE))/MASS) - GRAV
    SET VVELOC = VVELOC + VACCEL
    SET VDIST  = VDIST + VVELOC
    SET HACCEL = (FORCE * SIN(ANGLE))/MASS
```

```
SET VACCEL = ((FORCE * COS(ANGLE))/MASS) - GRAV
SET VVELOC = VVELOC + VACCEL
SET VDIST  = VDIST + VVELOC
SET HACCEL = (FORCE * SIN(ANGLE))/MASS
SET HVELOC = HVELOC + HACCEL
SET HDIST  = HDIST + HVELOC

SET TIME = TIME + 1

IF TIME > MAXT
THEN
  SET FLAG = 1
ELSE

SET TIME = TIME - 1

PRINT  MASS, VACCEL,
VVELOC, VDIST, HACCEL,
HVELOC, HDIST

END OF ROCKET
```


PROGRAM ROCKET
SET VACCEL = 0
SET VVELOC = 0
SET VDIST = 0
SET HACCEL = 0
SET HVELOC = 0
SET HDIST = 0
SET ANGLE = 0
SET TILT = 0.3491
SET GRAV = 32
SET MASS = 10000
SET FUEL = 50
SET FORCE = 400000
READ FROM 'MAX': MAXT
SET FLAG = 0
SET TIME = 1
DO WHILE FLAG = 0
IF TIME ≤100
THEN
ELSE
SET VACCEL = ((FORCE * COS(ANGLE))/MASS) - GRAV
SET VVELOC = VVELOC + VACCEL
SET VDIST = VDIST + VVELOC
SET HACCEL = (FORCE * SIN(ANGLE))/MASS
SET HVELOC = HVELOC + HACCEL
SET HDIST = HDIST + HVELOC
SET TIME = TIME + 1
SET MASS = MASS-FUEL
IF TIME = 11
THEN
ELSE
SET ANGLE = TILT
IF TIME = 101
THEN
ELSE
SET FORCE = 0

PDL — HIERARCHICAL



APPENDIX C DETAILED ERROR ANALYSIS BY PROGRAM

PROGRAM

TYPE OF ERROR	INVENTORY	ROCKET	AIRPORT	TOTAL
INCORRECT LOCATION FOR STATEMENT	21 (66%)	13 (24%)	8 (15%)	42 (30%)
STATEMENT MISSING	7 (22%)	9 (16%)	18 (33%)	34 (24%)
INCORRECT/MISSING VARIABLE	1 (3%)	11 (20%)	0	12 (8%)
INCORRECT LOGICAL OPERATOR	0	3 (5%)	4 (8%)	7 (5%)
INCORRECT OR MISSING FORMAT OR STATEMENT LABEL	0	8 (15%)	7 (13%)	15 (11%)
EXTRA (ERRONEOUS) STATEMENT INCLUDED	0	0	5 (9%)	5 (4%)
SYNTAX	3 (9%)	11 (20%)	12 (22%)	26 (18%)
TOTAL	32 (100%)	55 (100%)	54 (100%)	141(100%)

TECHNICAL REPORTS DISTRIBUTION LIST

OFFICE OF NAVAL RESEARCH

Code 442

TECHNICAL REPORTS DISTRIBUTION LIST

OSD

CDR Paul R. Chatelier
Office of the Deputy Under Secretary of Defense
OUSDRE (E&LS)
Pentagon, Room 3D129
Washington, D.C. 20301

Department of the Navy

Engineering Psychology Programs
Code 442
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217 (5 cys)

Director
Communication & Computer Technology
Code 240
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Director
Manpower, Personnel and Training
Code 270
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Information Systems Program

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Physiology Program
Code 441
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Special Assistant for Marine Corps Matters
Code 100M
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Department of the Navy

Commanding Officer
ONR Eastern/Central Regional Office
ATTN: Dr. J. Lester
Building 114, Section D
666 Summer Street
Boston, MA 02210

Commanding Officer
ONR Branch Office
ATTN: Dr. C. Davis
1030 East Green Street
Pasadena, CA 91106

Commanding Officer
ONR Western Regional Office
ATTN: Dr. E. Gloye
1030 East Green Street
Pasadena, CA 91106

Office of Naval Research
Scientific Liaison Group
American Embassy, Room A-407
APO San Francisco, CA 96503

Director
Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375 (6 cys)

Dr. Robert G. Smith
Office of the Chief of Naval Operations, OP987H
Personnel Logistics Plans
Washington, D.C. 20350

Dr. Jerry C. Lamb
Combat Control Systems
Naval Underwater Systems Center
Newport, RI 02840

Naval Training Equipment Center
ATTN: Technical Library
Orlando, FL 32813

Department of the Navy

Human Factors Department
Code N215
Naval Training Equipment Center
Orlando, FL 32813

Dr. Alfred F. Smode
Training Analysis and Evaluation Group
Naval Training Equipment Center
Code N-OOT
Orlando, FL 32813

Mr. Louis Chmura
Code 7592
Naval Research Laboratory
Washington, DC 20375

Dr. Gary Poock
Operations Research Department
Naval Postgraduate School
Monterey, CA 93940

Dean of Research Administration
Naval Postgraduate School
Monterey, CA 93940

Mr. Warren Lewis
Human Engineering Branch
Code 8231
Naval Ocean Systems Center
San Diego, CA 92152

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

J. B. Blankenheim
Code 47013
Naval Electronics Systems Command
NC Bldg. #1, Room 4E40
Washington, DC 20360

Commanding Officer
MCTSSA
Marine Corps Base
Camp Pendleton, CA 92055

Department of the Navy

Mr. Arnold Rubinstein
Naval Material Command
NAVMAT 0722 - Rm. 508
800 North Quincy Street
Arlington, VA 22217

Commander
Naval Air Systems Command
Human Factors Programs
NAVAIR 340F
Washington, D.C. 20361

Commander
Naval Air Systems Command
Crew Station Design,
NAVAIR 5313
Washington, D.C. 20361

Mr. Phillip Andrews
Naval Sea Systems Command
NAVSEA 0341
Washington, D.C. 20362

Commander
Naval Electronics Systems Command
Human Factors Engineering Branch
Code 4701
Washington, D.C. 20360

Mr. John Impagliazzo
Code 101
Newport Laboratory
Naval Underwater Systems Center
Newport, RI 02840

CDR Robert Biersner
Naval Medical R&D Command
Code 44
Naval Medical Center
Bethesda, MD 20014

Dr. Arthur Bachrach
Behavioral Sciences Department
Naval Medical Research Institute
Bethesda, MD 20014

Dr. George Moeller
Human Factors Engineering Branch
Submarine Medical Research Lab
Naval Submarine Base
Groton, CT 06340

Department of the Navy

Dr. Mel C. Moy
Code 302
Nava' Personnel R&D Center
Sa: .ego, CA 92152

Dr. Richard Neetz
Code 1226
Pacific Missile Test Center
Pt Mugu, CA 93042

Navy Personnel Research and Development Center
Planning & Appraisal
Code 04
San Diego, CA 92152

Navy Personnel Research and Development Center
Management Systems, Code 303
San Diego, CA 92152

Navy Personnel Research and Development Center
Performance Measurement & Enhancement
Code 309
San Diego, CA 92152

Dr. Julie Hopson
Human Factors Engineering Division
Naval Air Development Center
Warminster, PA 18974

Mr. Jeffrey Grossman
Human Factors Branch
Code 3152
Naval Weapons Center
China Lake, CA 93555

Human Factors Engineering Branch
Code 1226
Pacific Missile Test Center
Point Mugu, CA 93042

Mr. J. Williams
Department of Environmental Sciences
U.S. Naval Academy
Annapolis, MD 21402

Department of the Navy

Dean of the Academic Departments
U.S. Naval Academy
Annapolis, MD 21402

Human Factors Section
Systems Engineering Test Directorate
U.S. Naval Air Test Center
Patuxent River, MD 20670

Human Factor Engineering Branch
Naval Ship Research and Development Center, Annapolis Division
Annapolis, MD 21402

CDR W. Moroney
Code 55MP
Naval Postgraduate School
Monterey, CA 93940

Mr. Merlin Malehorn
Office of the Chief of Naval Operations (OP-115)
Washington, D.C. 20350

Department of the Army

Mr. J. Barber
HQS, Department of the Army
DAPE-MBR
Washington, D.C. 20310

Dr. Joseph Zeidner
Technical Director
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Director, Organizations and Systems Research Laboratory
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Technical Director
U.S. Army Human Engineering Labs
Aberdeen Proving Ground, MD 21005

Department of the Army

ARI Field Unit-USAREUR
ATTN: Library
C/O ODCSPER
HQ USAREUR & 7th Army
APO New York 09403

Department of the Air Force

U.S. Air Force Office of Scientific Research
Life Sciences Directorate, NL
Bolling Air Force Base
Washington, D.C. 20332

Chief, Systems Engineering Branch
Human Engineering Division
USAF AMRL/HES
Wright-Patterson AFB, OH 45433

Air University Library
Maxwell Air Force Base, AL 36112

Dr. Earl Alluisi
Chief Scientist
AFHRL/CCN
Brooks AFB, TX 78235

Foreign Addressees

North East London Polytechnic
The Charles Myers Library
Livingstone Road
Stratford
London E15 2LJ
ENGLAND

Professor Dr. Carl Graf Hoyos
Institute for Psychology
Technical University
8000 Munich
Arcisstr 21
FEDERAL REPUBLIC OF GERMANY

Dr. Kenneth Gardner
Applied Psychology Unit
Admiralty Marine Technology Establishment
Teddington, Middlesex TW11 OLN
ENGLAND

Foreign Addressees

Director, Human Factors Wing
Defence & Civil Institute of Environmental Medicine
Post Office Box 2000
Downsview, Ontario M3M 3B9
CANADA

Dr. A. D. Baddeley
Director, Applied Psychology Unit
Medical Research Council
15 Chaucer Road
Cambridge, CB2 2EF
ENGLAND

Other Government Agencies

Defense Technical Information Center
Cameron Station, Bldg. 5
Alexandria, VA 22314 (12 cys)

Dr. Craig Fields
Director, Cybernetics Technology Office
Defense Advanced Research Projects Agency
1400 Wilson Blvd
Arlington, VA 22209

Other Organizations

Dr. H. McI. Parsons
Human Resources Research Office
300 N. Washington Street
Alexandria, VA 22314

Dr. Jesse Orlansky
Institute for Defense Analyses
400 Army-Navy Drive
Arlington, VA 22202

Dr. Arthur I. Siegel
Applied Psychological Services, Inc.
404 East Lancaster Street
Wayne, PA 19087

Dr. Robert T. Hennessy
NAS - National Research Council
2101 Constitution Avenue, N.W.
Washington, DC 20418

Other Organizations

Dr. Timothy Lindquist
Department of Computer Science
VPI & SU
Blacksburg, VA 24061

Dr. M. G. Samet
Perceptronics, Inc.
6271 Variel Avenue
Woodland Hills, CA 91364

Dr. Robert Williges
Human Factors Laboratory
Virginia Polytechnical Institute
and State University
130 Whittemore Hall
Blacksburg, VA 24061

Mr. Edward M. Connelly
Performance Measurement
Associates Inc.
410 Pine Street, S.E.
Suite 300
Vienna, VA 22180

DAT

FILM